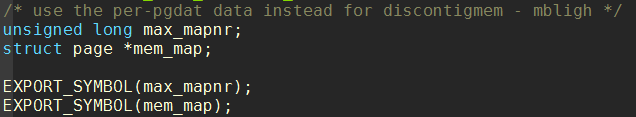
对内存的管理，分为两个部分：物理内存的管理、虚拟内存的管理

物理内存的管理主要涉及硬件资源，即对RAM的管理，虚拟内存的管理指对进程地址空间的管理，物理内存和虚拟内存间的关系通过page fault联系起来。

对RAM的管理，主要包括页框管理和内存区管理。（什么是内存区管理？？）

对虚拟内存的管理，包括内存区管理、内存映射、伙伴系统、slab和内存池。

内核使用mem\_map页框描述符数组，来存放所有的页框描述符，进而管理所有的页框。



几个概念：

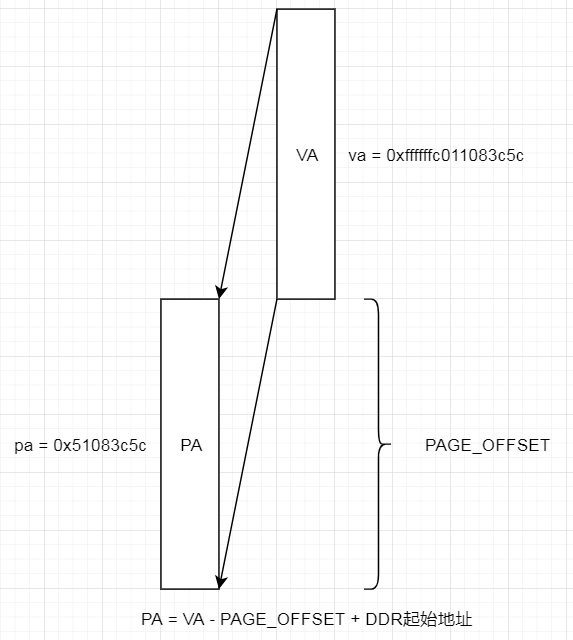
虚拟地址、物理地址、页框描述符、页框号（物理页号）

虚拟地址 -> 物理地址：virt\_to\_phys

1. #define virt\_to\_phys virt\_to\_phys
2. **static** **inline** phys\_addr\_t virt\_to\_phys(**const** **volatile** **void** \*x)
3. {
4. **return** \_\_virt\_to\_phys((unsigned **long**)(x));
5. }
6. #define \_\_virt\_to\_phys(x)   (((phys\_addr\_t)(x) - PAGE\_OFFSET + PHYS\_OFFSET))
8. /\* PAGE\_OFFSET是物理地址和虚拟地址间的位移量，也即是虚拟地址能拥有的最大值
9. \* 页大小为4Kb，所以页内偏移为12位
10. \* 页大小为4Kb，一个页表为64位(8 byte)，所以一个页表最多可以存放2^9页表项，即PGD或PMD为9位
11. \* 三级页表，所以虚拟地址位数为9+9+9+12=39
12. \*/
13. #define CONFIG\_ARM64\_VA\_BITS 39
14. #define VA\_BITS         (CONFIG\_ARM64\_VA\_BITS)
15. #define PAGE\_OFFSET     (UL(0xffffffffffffffff) << (VA\_BITS - 1))
17. /\* PHYS\_OFFSET是DDR的物理起始地址 \*/
18. #define PHYS\_OFFSET     ({ memstart\_addr; })

为什么要减去一个 PAGE\_OFFSET：

内核物理地址和虚拟地址间使用直接映射，PAGE\_OFFSET与虚拟地址实际使用的位数有关，代表整个内核空间的起始地址（比这小的地址都是用户空间地址）。



物理地址 -> 虚拟地址：phys\_to\_virt

和virt\_to\_phys实现过程相反，不再阐述。

1. #define phys\_to\_virt phys\_to\_virt
2. **static** **inline** **void** \*phys\_to\_virt(phys\_addr\_t x)
3. {
4. **return** (**void** \*)(\_\_phys\_to\_virt(x));
5. }
6. #define \_\_phys\_to\_virt(x)   ((unsigned long)((x) - PHYS\_OFFSET + PAGE\_OFFSET))

虚拟地址 -> 页框描述符(struct page)：virt\_to\_page

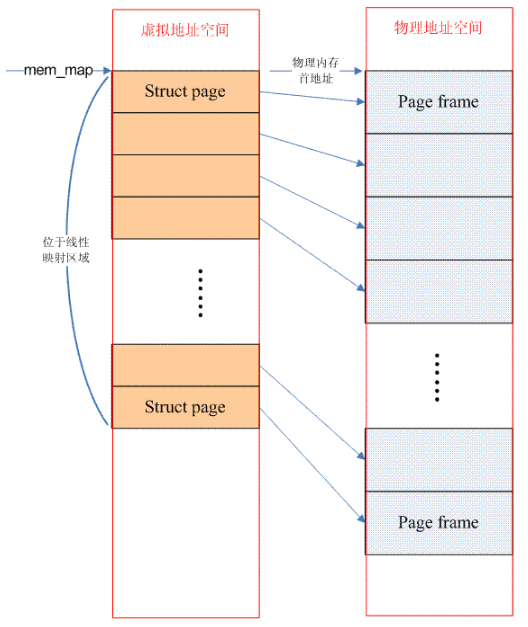
Linux的内存模型

参考wow大神的文章：flat memory、discontiguous memory、sparse memory

<http://www.wowotech.net/memory_management/memory_model.html>

flat memory：

物理地址空间是一块连续的、没有空洞的内存空间。



每一个page frame都有一个对应的struct page页框描述符。

所有的struct page由一个mem\_map数组进行管理，每一个数组条目指向一个实际的page frame。

1. unsigned **long** max\_mapnr;
2. **struct** page \*mem\_map;

flat memory中pfn到page的转换很简单，mem\_map基地址加上pfn为偏移即可，page到pfn的转换反之。

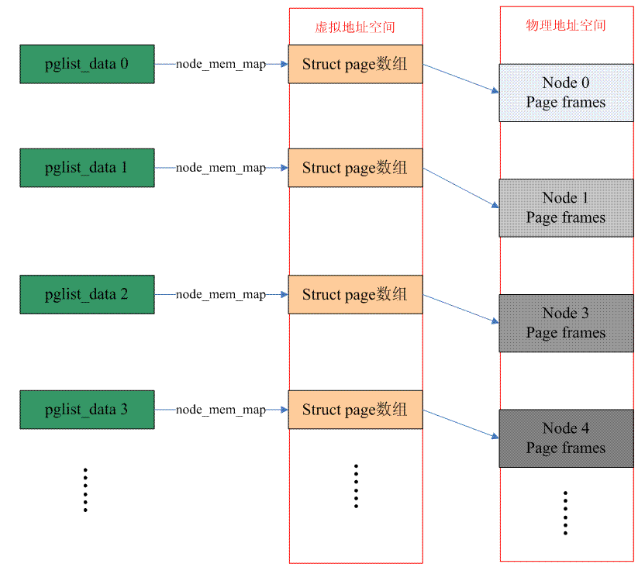
1. #define \_\_pfn\_to\_page(pfn)  (mem\_map + ((pfn) - ARCH\_PFN\_OFFSET))
2. #define \_\_page\_to\_pfn(page) ((unsigned long)((page) - mem\_map) + \
3. ARCH\_PFN\_OFFSET)

discontiguous memory：

如果物理地址空间有空洞，不连续，这种内存模型就是discontiguous memory model。

每一段被分割的物理地址空间，称之为一个node，每一个node内部有一个对应的node\_mem\_map数组对节点内部的页框进行管理。

即按照node进行划分。



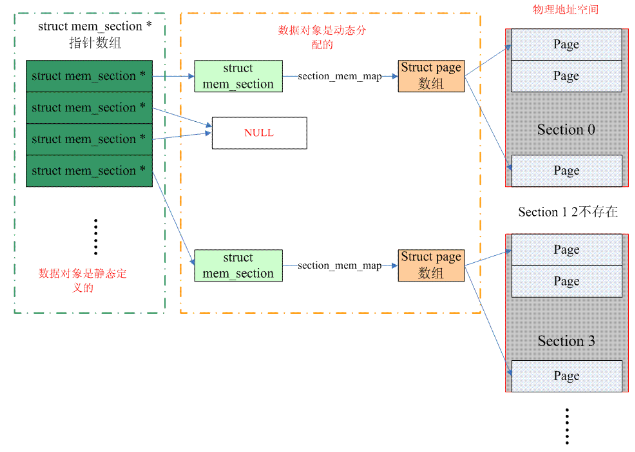
Discontiguous memory模型的pfn到page的转换分为两步：

pfn -> node\_id -> node\_mem\_map -> page

1. 根据pfn找到这个pfn对应的node\_id，即找到对应的node\_mem\_map数组；
2. 参考flat memory在node\_mem\_map中找到对应的struct page页框描述符。
3. #define \_\_pfn\_to\_page(pfn)          \
4. ({  unsigned **long** \_\_pfn = (pfn);        \
5. unsigned **long** \_\_nid = arch\_pfn\_to\_nid(\_\_pfn);  \
6. NODE\_DATA(\_\_nid)->node\_mem\_map + arch\_local\_page\_offset(\_\_pfn, \_\_nid);\
7. })

sparse memory：

为了解决内存热插拔的问题，地址空间按照section进行划分，每一个section都是可热插拔的内存。



sparse memory witch vmemmap：

在sparse memory上的一点改进，也是我们当前内核所使用的方法。